Attorney Docket No.: YOR920030249US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In Re PATENT APPLICATION Of:	
Jude A. Rivers	
Appln. No: 10/644,210	Art Unit: 2189
Filed: August 20, 2003	Examiner: Reba I. ELMORE
For: DISTRIBUTED BUFFER INTEGRATED CACHE MEMORY ORGANIZATION AND METHOD OF REDUCING ENERGY CONSUMPTION THEROF	

Mail Stop **APPEAL BRIEF – PATENTS**
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF UNDER 37 CFR §41.37(a)

Appellants have filed a timely Notice of Appeal, November 16, 2006, appealing a final Office action, dated June 16, 2006. A single copy of this brief is provided pursuant to 37 C.F.R. §41.37(a). An authorization to charge the appropriate fee is included herewith. Please charge any deficiencies in fees and credit any overpayment of fees to IBM Corporation Deposit Account No. 50-0510 and advise us accordingly.

I. Real Party in Interest:

International Business Machines Corporation is the real party in interest in the above referenced patent application.

II. Related Appeals and Interferences:

The appellant is aware of no other appeals or interferences that will directly affect or have a bearing on this appeal.

III. Status of The Claims:

Claims 1 – 29 are currently pending.

Claims 1 – 29 are currently finally rejected and are the subject of this appeal. All claims are appended in a "Claims Appendix" attached hereto.

IV. Status of Amendments:

All amendments have been entered and are reflected in the appended claims.

V. Summary of The Claimed Subject Matter:

Independent Claim 1

The invention, as recited by claim 1, is a cache memory 80A, B (or 100) that includes a storage array 86 (or 106) and a cache buffer 84 or 88 (or 108, 110 or 112). *See, e.g.*, page 5, line 28 – page 7, line 19 of the application with reference to: Figure 2A a copy of which is included as Exhibit A in the Evidence Appendix of this Appeal Brief; Figure 2B a copy of which is included as Exhibit B in the Evidence Appendix of this Appeal Brief; and Figure 3, a copy of which is included as Exhibit C in the Evidence Appendix of this Appeal Brief. The cache buffer(s) 84 and/or 88 (108, 110 and/or 112) contain(s) most recently accessed data. *See, e.g.*, page 6, lines 3 – 5 ("A CPU data request is first presented in parallel to the staging buffer 84 and the hollow buffer 88 (which holds a number (*i*) of the most recently accessed lines from the holding cache)"). The storage array 86, 106 selectively receives data from the cache buffer(s) 84, 88, 108, 110, 112. *See, e.g.*, page 6, lines 10 – 11 ("Upon a cache miss, the data from the staging buffer 84 is promoted into the holding cache 86 and the incoming missed data block 82 is brought into the staging buffer 84."). A tag memory 102 storing tags associated with data in the storage array 106. Tag memory 104 also stores tags for data in the cache buffer(s) 108, 110 and 112. *See, e.g.*, page 7, lines 5 – 12 ("Tags are stored in an *n*-location CAM or *n*-CAM 102 and an *i*-location CAM or *i*-CAM 104, … . The *n* tags in the *n*-CAM 102 are each associated with corresponding blocks in HC 106 and *i*-CAM tags being associated with corresponding blocks in HB 108.").

Dependent Claim 3

The cache buffer(s), as recited, by claim 3, includes a cache input buffer 110. *See, e.g.*, Figure 3.

Dependent Claim 4

The cache buffer(s), as recited, by claim 4, includes a cache output buffer 88, 108 with most recently accessed data. *See, e.g.*, page 6, lines 3 – 5 (“the hollow buffer 88 (which holds a number (*i*) of the most recently accessed lines from the holding cache)”).

Dependent Claim 5

The tag memory, as recited by claim 5, includes a first CAM 102 and second CAM 104. *See, e.g.*, page 7, lines 5 – 12 (“Tags are stored in an *n*-location CAM or *n*-CAM 102 and an *i*-location CAM or *i*-CAM 104, … . The *n* tags in the *n*-CAM 102 are each associated with corresponding blocks in HC 106 and *i*-CAM tags being associated with corresponding blocks in HB 108.”).

Dependent Claim 6

As recited by claim 6, the second CAM 104 is checked before the first CAM 102. *See, e.g.*, page 7, lines 21 – 28 (“Power is reduced in CAMRAM cache 100 over prior art CAMRAM caches … [because], the *i* most recently accessed cache data blocks, which have the highest likelihood of being requested in immediately subsequent accesses, are held in the HB 108. So, …, the incoming tag is compared against *i*-CAM entries, which are most likely to match.”).

Dependent Claim 9

As recited by claim 1, power is substantially reduced for cache buffer 84 and/or 88 (108, 110 and/or 112) accesses than for storage array 86, 106 accesses. *See, e.g.*, page 7, lines 21 – 28 (“Thus, finding a match in the *i*-CAM 104 saves power, CAM power that is otherwise expended searching *n*-CAM 102.”).

Independent Claim 10

The invention, as recited by claim 10, is a content addressable memory (CAM) random access memory (RAM) cache 130 with multiple CAMRAM banks 132. *See, e.g.*, page 9, lines 6 – 16 of the application with reference to Figure 4, a copy of which is included as Exhibit D in the Evidence Appendix of this Appeal Brief. Each CAMRAM bank 132 includes a cache buffer 108, 134, 136 that contains most recently accessed data and receives cache input data. *Id*, lines 8 – 10 ("the CSB includes a CSB line 134 in each bank 132. Also, a single SBB 136 serves all banks 132, receiving individual outputs 138, 140 from each bank HC 106 and HB 108."). A bank store 132 selectively receives data from the cache buffer 134 and 136. *See, e.g.*, page 9, lines 18 – 25 ("Cache line data are passed into the HC 106 during a dead cycle in a subsequent L1 cache miss, … . An *n*-CAM hit causes a copy of respective cache line to move from the HC 106 into the HB 108 and the corresponding HB 108 entry is invalidated."). A tag memory 102 storing tags associated with data in the storage array 106. A tag CAM 102, 104 also stores tags for data in the bank store and cache buffers 134 and 136. *See, e.g.*, page 7, lines 5 – 12 ("Tags are stored in an *n*-location CAM or *n*-CAM 102 and an *i*-location CAM or *i*-CAM 104, … . The *n* tags in the *n*-CAM 102 are each associated with corresponding blocks in HC 106 and *i*-CAM tags being associated with corresponding blocks in HB 108.").

Dependent Claim 11

The cache buffer, as recited, by claim 11, includes an input buffer line 134 and a cache output buffer 108 with most recently accessed data. *See, e.g.*, page 9, lines 8 – 9 ("the CSB includes a CSB line 134 in each bank 132."); *and see*, page 6, lines 3 – 5 ("the hollow buffer 88 (which holds a number (*i*) of the most recently accessed lines from the holding cache)").

Dependent Claim 13

The tag memory, as recited by claim 13, includes an *n*-CAM 102 and an *i*-CAM 104. *See, e.g.*, page 7, lines 5 – 12 ("Tags are stored in an *n*-location CAM or *n*-CAM 102 and an *i*-location CAM or *i*-CAM 104, … . The *n* tags in the *n*-CAM 102 are each associated with corresponding blocks in HC 106 and *i*-CAM tags being associated with corresponding blocks in HB 108.").

Dependent Claim 14

As recited by claim 14, the *i*-CAM 104 is checked before the *n*-CAM 102. *See, e.g.*, page 7, lines 21 – 28 ("Power is reduced in CAMRAM cache 100 over prior art CAMRAM caches … [because], the *i* most recently accessed cache data blocks, which have the highest likelihood of being requested in immediately subsequent accesses, are held in the HB 108. So, …, the incoming tag is compared against *i*-CAM entries, which are most likely to match.").

Dependent Claim 16

As recited by claim 16, power is substantially reduced for cache buffer 84 and/or 88 (108, 110 and/or 112) accesses than for storage array 86, 106 accesses. *See, e.g.*, page 7, lines 21 – 28 ("Thus, finding a match in the *i*-CAM 104 saves power, CAM power that is otherwise expended searching *n*-CAM 102.").

Independent Claim 19

The invention, as recited by claim 19, is a method of managing data in a cache. *See, e.g.*, page 9, line 26 – page 11, line 18 of the application with reference to Figure 5 (loading from) and 6 (storing to), copies of which are included as Exhibit E and F, respectively, in the Evidence Appendix of this Appeal Brief. Incoming data is provided

to an input buffer 190. Loaded data may be moved from the input buffer into the storage array 204. Data from the storage array is selectively loaded to an output buffer 170s. A number of most recently accessed data blocks are held in the output buffer. Data from each of the input buffer, the storage array and the output buffer are selectively provided 162 in response to an access request 152.

Dependent Claim 20

As recited by claim 20, the method further includes receiving an access request for data 152, 182 and checking the input data buffer 156, 186 for data requested for access. *See, e.g.*, Figures 5 and 6.

Dependent Claim 21

As recited by claim 21, the method further includes storing said data in said input buffer 190 and marking said stored data as dirty. *See, e.g.*, Figure 6.

Dependent Claim 22

As recited by claim 22, the method further includes checking the output buffer 160, 192 for data requested for access. *See, e.g.*, Figures 5 and 6.

Dependent Claim 25

As recited by claim 25, wherein whenever requested data is not found in the output or input buffer the storage array is checked 166, 198 for data requested for access. *See, e.g.*, Figures 5 and 6.

VI. Grounds of Rejection to Be Reviewed on Appeal:

Claims 1 – 29 are finally rejected under 35 U.S.C. §102(e) over published U.S. Patent Application No. 2004/0010675 to Moritz.

VII. Argument:

Moritz fails, both literally and inherently, to disclose specifically recited features of claims 1 – 29. Moritz and the present application are clearly different, both by the representation in the Figures of each and by the plain language of each. Specifically, Moritz fails to teach a cache memory with cache storage buffered by a cache buffer that contains "most recently accessed data," as claims 1 and 10 recite; and, Moritz fails to teach "selectively loading accessed data from said storage array to an output buffer, a number of most recently accessed data blocks being held in said output buffer" as claim 19 recites. Therefore, Moritz does not teach the present invention as recited in claims 1, 10 or 19 or any claims depending therefrom.

"A claim is anticipated only if each and every element **as set forth in the claim** is found, either expressly or inherently described, in a single prior art reference."[1] "The identical invention must be shown in as complete detail as is contained in the ... claim."[2] Furthermore, "[d]uring patent examination, the pending claims must be 'given *>their< broadest **reasonable** interpretation consistent with the specification.' >*In re Hyatt*, 211 F.3d 1367, 1372, 54 USPQ2d 1664, 1667 (Fed. Cir. 2000)"[3] Moreover, that "broadest reasonable interpretation of the claims **must also be consistent** with the interpretation that those skilled in the art would reach. *In re Cortright*, 165 F.3d 1353, 1359, 49 USPQ2d 1464, 1468 (Fed. Cir. 1999)."[4]

Generally, it has been asserted that Moritz Figure 4 (a copy of which is included as Exhibit G in the Evidence Appendix of this Appeal Brief) anticipates the present invention, primarily, with reference to paragraphs 0105 – 0109. Moritz Figure 4 shows four (4) storage arrays (Data Arrays at Way 0 – 3) with everything else directed to

1 *Verdegaal Bros. v. Union Oil Co. of California*, 814 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987) (emphasis added).
2 *Richardson v. Suzuki Motor Co.*, 868 F.2d 1226, 1236, 9 USPQ2d 1913, 1920 (Fed. Cir. 1989).
3 MPEP, §2111 (emphasis added).
4 *Id*, (emphasis added).

addressing these 4 storage arrays in parallel. Certainly, one does not access one array through another. None of these 4 storage arrays are shown as being buffered before or after the array. Neither does Moritz describe them as being buffered. Nor is any of the four acting as a buffer for any other of the four.

Moritz does not describe or suggest that these 4 storage arrays (Data Arrays at Way 0 – 3) communicate with one another. However, to read the present claims on Moritz, such that one "storage array … selectively [receives] data from said cache buffer (one of the other storage arrays)," there must be communication between at least two of the four. Without communication between the recited storage array and cache buffer of claim 1, for example, one (i.e., the array or the buffer) cannot receive anything from the other (the buffer or the array) and vice versa. Therefore, communication is necessary to label one as a buffer and the other as the array. However, there is no communication between the four Moritz storage arrays.

Further, since the cache buffer may include an input buffer (*See*, claims 3, 11 and 19 – 21.) and an output buffer (*See*, claims 4, 11 and 19.), each buffering the storage array; to assert that Moritz shows this, there must be communication between at least three of the four Moritz storage arrays. Otherwise, an input buffer that has no communications path with what it is to buffer is not a buffer. Neither is an output buffer. Therefore, Moritz fails to teach or suggest storage arrays communicating with one another and, in particular, "a storage array … selectively receiving data from said cache buffer," as recited in claims 1, 3, 4, 10, 11 and 19 – 21.

Moritz also provides that the Tag-cache 210 is different than both the cache buffer(s) and the recited tag memory/memories. At the end of paragraph 0118, Moritz specifically indicates that "each Tag-Cache 210 entry is exactly the same as a hotline register 208, and performs the same functions, but dynamically." Therefore, a Moritz Tag-Cache 210 entry, like the

> hotline register 208 has 3 components: (1) protection bits (ASID), which are used to enforce address space protection, (2) TagIndex--two accesses are to the same cache line if their Tag and Index components are the same. The TagIndex component is compared with Tag and Index of the actual access to check if the hotline register can indeed be used to directly address the cache, (3) cache-way information--this information enables direct access to one of the ways in the set-associative cache.

Paragraph 0115. This is quite different than both the cache buffer(s) and the tag memory of the present application.

The Moritz Tag-cache 210 is not a cache buffer and, certainly not a buffer before (an input buffer) or after (an output buffer) the 4 storage arrays. While both the Tag-Cache 210 and the hotline register 208 store cache line addresses for data in the Moritz data arrays, neither stores "most recently accessed data" as recited in any of the claims 1 – 29. Further, even if one were to accept *arguendo*, that the Moritz Tag-Cache 210 were a cache buffer within a reasonable interpretation of the present specification[5], Moritz fails to provide anything, in describing either the Moritz Tag-Cache 210 or the Moritz hotline register 208, that could be considered as "storing tags associated with … selected data in said cache buffer" as recited in claims 1 and 10 recite. Certainly, the Moritz Hotline MISS line is not a tag "associated with … selected data in said cache buffer." *Id.* Neither is the Moritz translation buffer which contains address translations, an input buffer or an output buffer.

Furthermore, Moritz does not teach a tag memory that may include two tag memories, e.g., as recited in claims 5, 6 and 13. Specifically,

> The subject architecture in the 2nd embodiment extends associative cache lookup mechanism 215, 216, 217, 218, with simpler, **direct addressing** modes 213, in a **virtually** tagged and indexed cache organization. This direct addressing mechanism 213 **eliminates** the associative **tag-checks** (i.e., **no tag-lookup** as shown in 215, 216, 217, 218 is required) and data-array accesses (i.e., only one of the data-arrays from 215, 216, 217, 218 is accessed).

5 *Supra*.

Moritz Paragraph 0105 (emphasis added). By contrast claim 1 (with analogous recitations in claims 10 and 19), for example, recites “a tag memory storing tags **associated** with data in said storage array” at lines 7 – 8 (emphasis added). Moreover, a cache buffer that contains “most recently accessed data,” as the claims recite, is quite different than storing “cache line **addresses** for the most recently accessed cache lines” in the four Moritz storage arrays as Moritz teaches.

Nowhere does Moritz describe an i-CAM and/or an n-CAM, for example, with one (the i-CAM) being directed to a buffer with most recently accessed data. *See*, claims 5, 6, 14, 20 – 22 and 25. As the claims recite, the i-CAM points to output buffer locations, where the most recently accessed data is cloistered. By cloistering these most recently accessed data in the output buffer and first checking just the tags for the output buffer, substantial power savings are realized as recited in claims 9 and 16. However, that is neither what Moritz teaches or suggests.

Furthermore, because dependent claims include all of the differences with the cited reference as the claims from which they depend, the present invention as recited in claims 2 – 8, 11 – 18 and 20 – 29, is neither taught, nor suggested by Moritz, alone or, further in combination with any reference of record.

CONCLUSION

Accordingly, because Moritz fails to teach a cache memory with a buffered storage array substantially as recited in claims 1 – 29, Moritz does not anticipate the present invention. The Appellants respectfully request that the board reverse the final rejection of claims 1 – 29 under 35 U.S.C. §102(3) over Moritz and pass the application to issue.

Respectfully submitted,

January 16, 2007
(Date)

/Charles W. Peterson, Jr., #34,406/
Charles W. Peterson, Jr.
Registration No. 34,406

Customer Number 33233
Law Office of Charles W. Peterson, Jr.
Suite 100
11703 Bowman Green Dr.
Reston, VA 20190
Telephone: (703) 481-0532
Facsimile: (703) 659-1485

CLAIMS APPENDIX

A copy of the claims involved in the appeal is provided below.

1. A cache memory comprising:

a cache buffer containing most recently accessed data;

a storage array comprising a plurality of cache memory locations and selectively receiving data from said cache buffer, selectively received said data being stored in ones of said memory locations; and

a tag memory storing tags associated with data in said storage array and selected data in said cache buffer.

2. A cache memory as in claim 1, wherein cache input data selectively includes executable commands.

3. A cache memory as in claim 1, wherein said cache buffer comprises:

a cache input buffer receiving cache input data.

4. A cache memory as in claim 3, wherein said cache buffer further comprises:

an output buffer, ones of said tags in said tag memory associated with said most recently accessed data in said output buffer.

5. A cache memory as in claim 4, wherein said tag memory comprises:

a first content addressable memory (CAM) containing tags associated with data stored in said storage array; and

a second CAM containing tags associated with said most recently accessed data.

6. A cache memory as in claim 5, wherein a tag for requested data is checked against tags in said second CAM and said cache input buffer before checking tags in said first CAM.

7. A cache memory as in claim 5, wherein each of said first CAM and said second CAM are a circulating first in first out register (FIFO).

8. A cache memory as in claim 4, wherein each said storage array is a static random access (SRAM) array.

9. A cache memory as in claim 1, wherein cache power is substantially less for accessing said data in said cache buffer than for accessing data in said storage array.

10. A content addressable memory (CAM) random access memory (RAM) cache comprising a plurality of CAMRAM banks, each of said CAMRAM banks comprising:

a cache buffer containing most recently accessed data and receiving cache input data, said cache input data selectively including executable commands;

a bank store comprising a plurality of cache memory locations and selectively receiving data from said cache buffer, selectively received said data being stored in ones of said memory locations; and

a CAM storing tags associated with data in said storage array and selected data in said cache buffer.

11. A CAMRAM as in claim 10, wherein said cache buffer comprises:

an input buffer line receiving a cache input data line; and

an output buffer containing said most recently accessed data, ones of said tags in said CAM being associated with said most recently accessed data.

12. A CAMRAM as in claim 11, further comprising a cache storage buffer, each said input buffer line in said plurality of CAMRAM banks being a line in said cache storage buffer.

13. A CAMRAM as in claim 11, wherein said CAM comprises:

an n-CAM having n tag locations, each n-CAM tag location being associated with one of n storage locations in said bank store; and

an i-CAM containing i tag locations, wherein n > i and each i-CAM tag location is associated with a location in said output buffer.

14. A CAMRAM as in claim 13, further comprising means for checking a tag for requested data against tags in said i CAM and said cache input buffer independent of tags in said n CAM.

15. A CAMRAM as in claim 14, wherein said checking means only checks for said tag in said n CAM, when said tag is not found in said i CAM or in said cache input buffer.

16. A CAMRAM as in claim 15, wherein cache power is substantially less for accessing said data in said cache buffer than for accessing data in said bank store.

17. A CAMRAM as in claim 13, wherein each of said n-CAM and said i-CAM are a circulating first in first out register (FIFO).

18. A CAMRAM as in claim 11, wherein said bank store is a static random access (SRAM) array.

19. A method of managing data in a cache, said method comprising the steps of:

a) providing incoming data to an input buffer;

b) selectively loading data from said input buffer into a storage array;

c) selectively loading accessed data from said storage array to an output buffer, a number of most recently accessed data blocks being held in said output buffer; and

d) selectively providing data from each of said input buffer, said storage array and said output buffer responsive to an access request.

20. A method of managing data as in claim 19, said method further comprising the steps of:

e) receiving an access request for data; and

f) checking said input data buffer for data requested for access.

21. A method of managing data as in claim 20, wherein said access request is a store request and said method further comprises the steps of:

g) storing said data in said input buffer; and

h) marking said stored data as dirty.

22. A method of managing data as in claim 20, said method further comprising the steps of:

g) checking said output buffer for said data requested for access.

23. A method of managing data as in claim 22, wherein said access request is a store request and said method further comprises the steps of:

h) storing said data in said output buffer; and

i) marking said stored data as dirty.

24. A method of managing data as in claim 22, wherein said output buffer is checked in step (g) coincident with checking said input buffer in step (f).

25. A method of managing data as in claim 22, wherein whenever said data requested for access is not found in said output buffer or said input buffer, said method further comprises the steps of:

h) checking said storage array for said data requested for access.

26. A method of managing data as in claim 25, wherein whenever said data requested for access is found in said storage array, said method further comprises the steps of:

i) loading said data requested for access into said output buffer; and

j) providing said data requested for access as an output.

27. A method of managing data as in claim 25, wherein whenever said data requested for access is not found in said storage array, said method further comprises the steps of:

i) sending a miss request;

j) loading said input buffer; and

k) providing said data from said input buffer as an output.

28. A method of managing data as in claim 27, wherein whenever said input buffer contains data other than said data requested for access, said sending step (h) further comprises loading other said data from input buffer to said output buffer.

29. A method of managing data as in claim 25, wherein data in each of said input buffer, said storage array and said output buffer are identified by tags, said tags being checked in checking steps (f), (g) and (h).

EVIDENCE APPENDIX

This section lists evidence submitted pursuant to 35 U.S.C. §§1.130, 1.131, or 1.132, or any other evidence entered by the Examiner and relied upon by Appellant in this appeal, and provides for each piece of evidence a brief statement setting forth where in the record that evidence was entered by the Examiner. Copies of each piece of evidence are provided as required by 35 U.S.C. §41.37(c)(ix).

Exhibit	EVIDENCE	BRIEF STATEMENT SETTING FORTH WHERE IN THE RECORD THE EVIDENCE WAS ENTERED BY THE EXAMINER
A	**Application Figure 2A**	**Originally filed with the application on August 20, 2003**
B	**Application Figure 2B**	**Originally filed with the application on August 20, 2003**
C	**Application Figure 3**	**Originally filed with the application on August 20, 2003**
D	**Application Figure 4**	**Originally filed with the application on August 20, 2003**
E	**Application Figure 5**	**Originally filed with the application on August 20, 2003**
F	**Application Figure 6**	**Originally filed with the application on August 20, 2003**
G	**Moritz Figure 4**	**Cited in the first Office action mailed December 19, 2005**
H		
I		
J		
K		
L		

EXHIBIT A

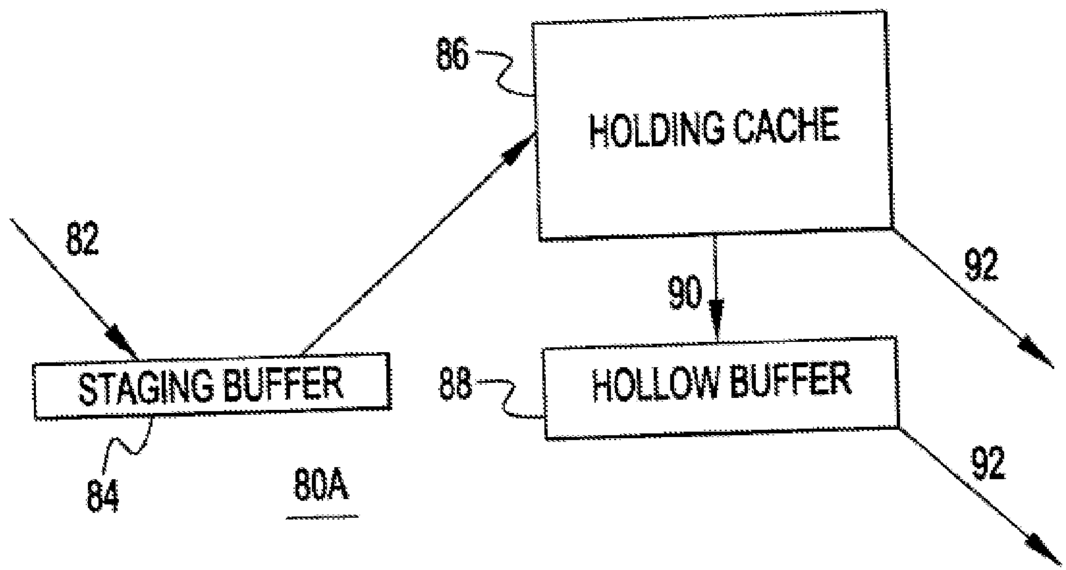

FIG.2A

EXHIBIT B

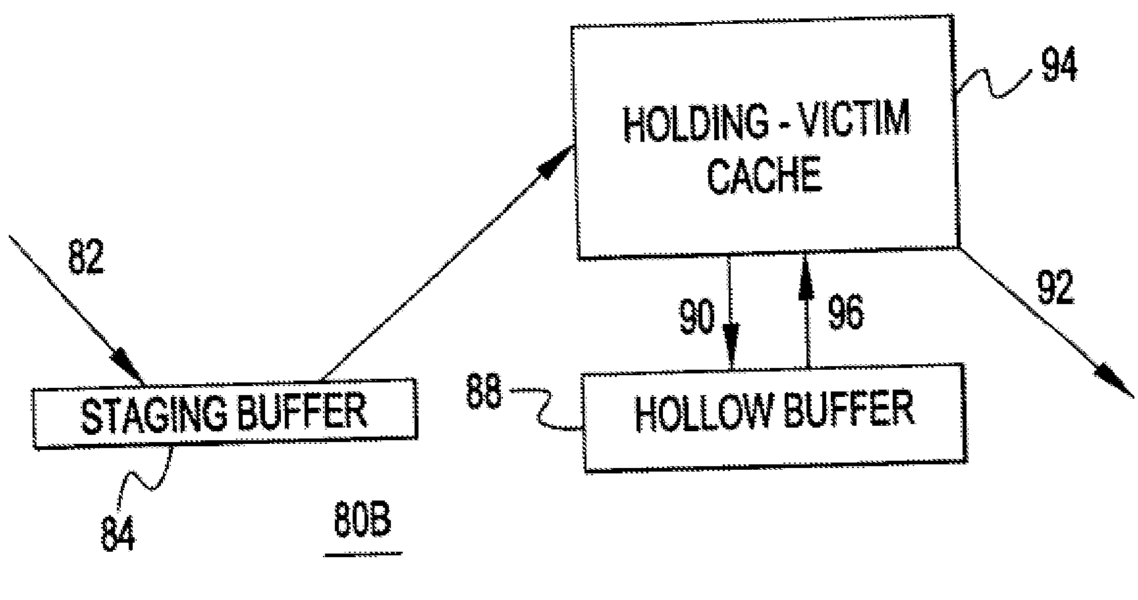

FIG.2B

EXHIBIT C

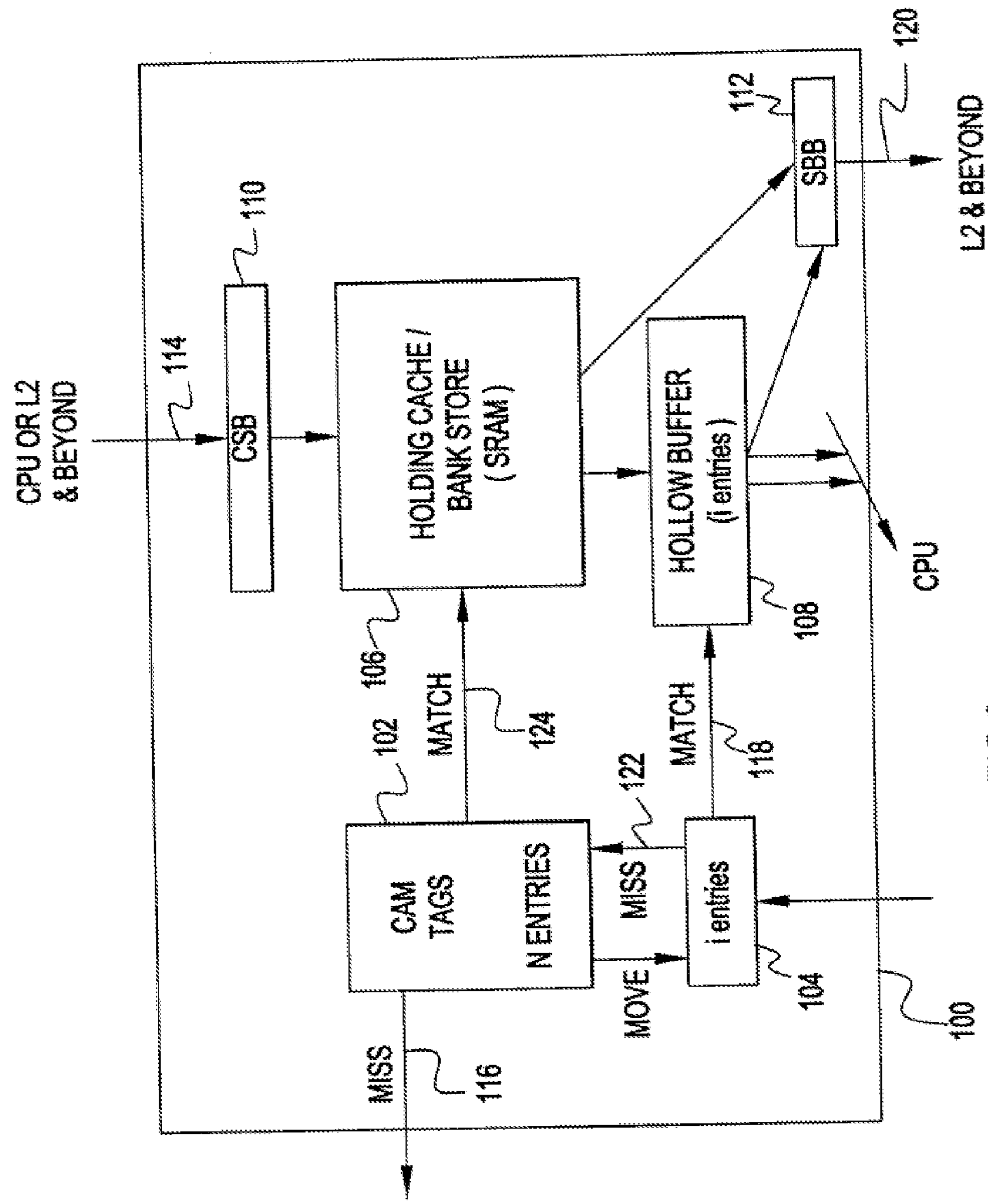

FIG.3

EXHIBIT D

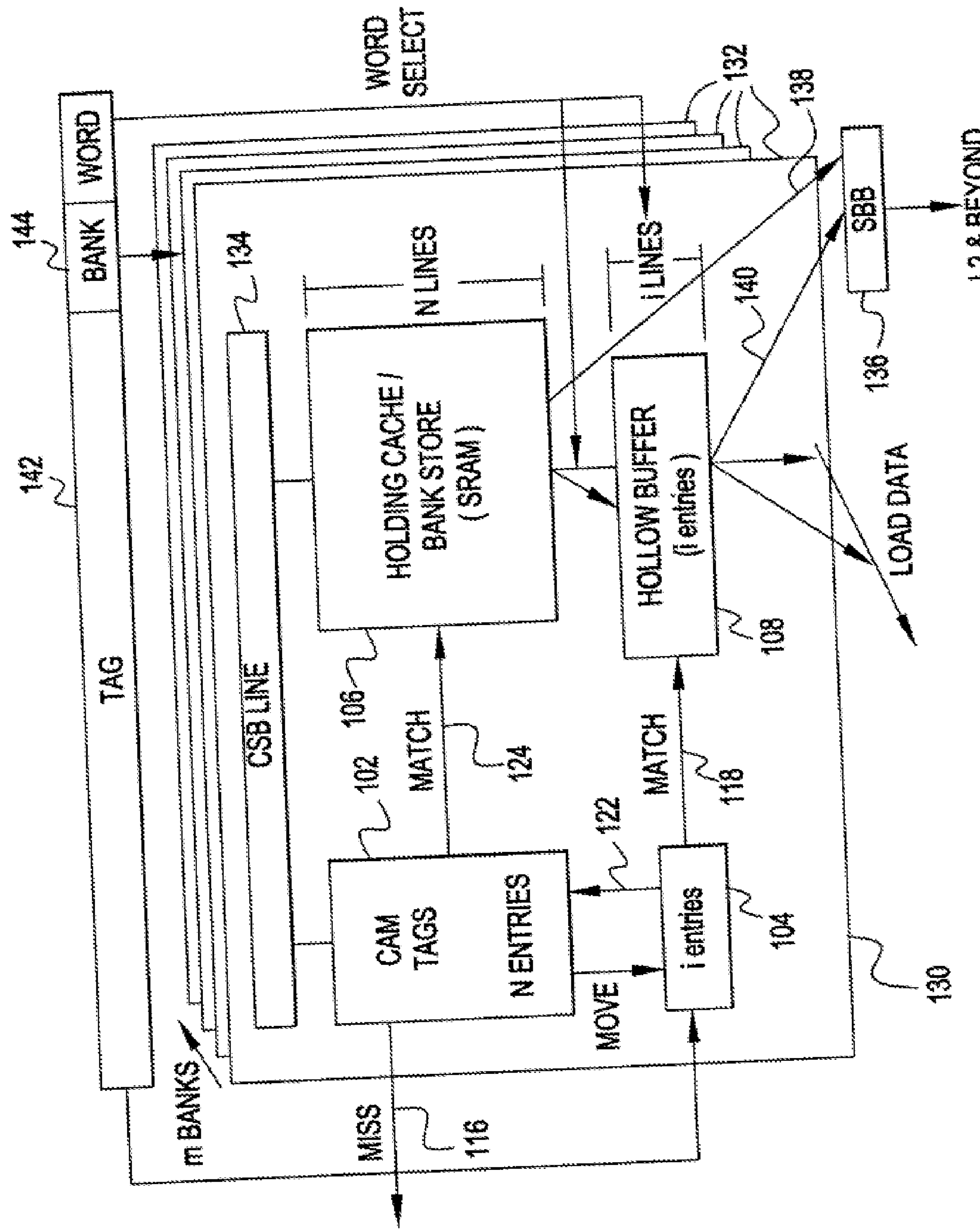

FIG.4

EXHIBIT E

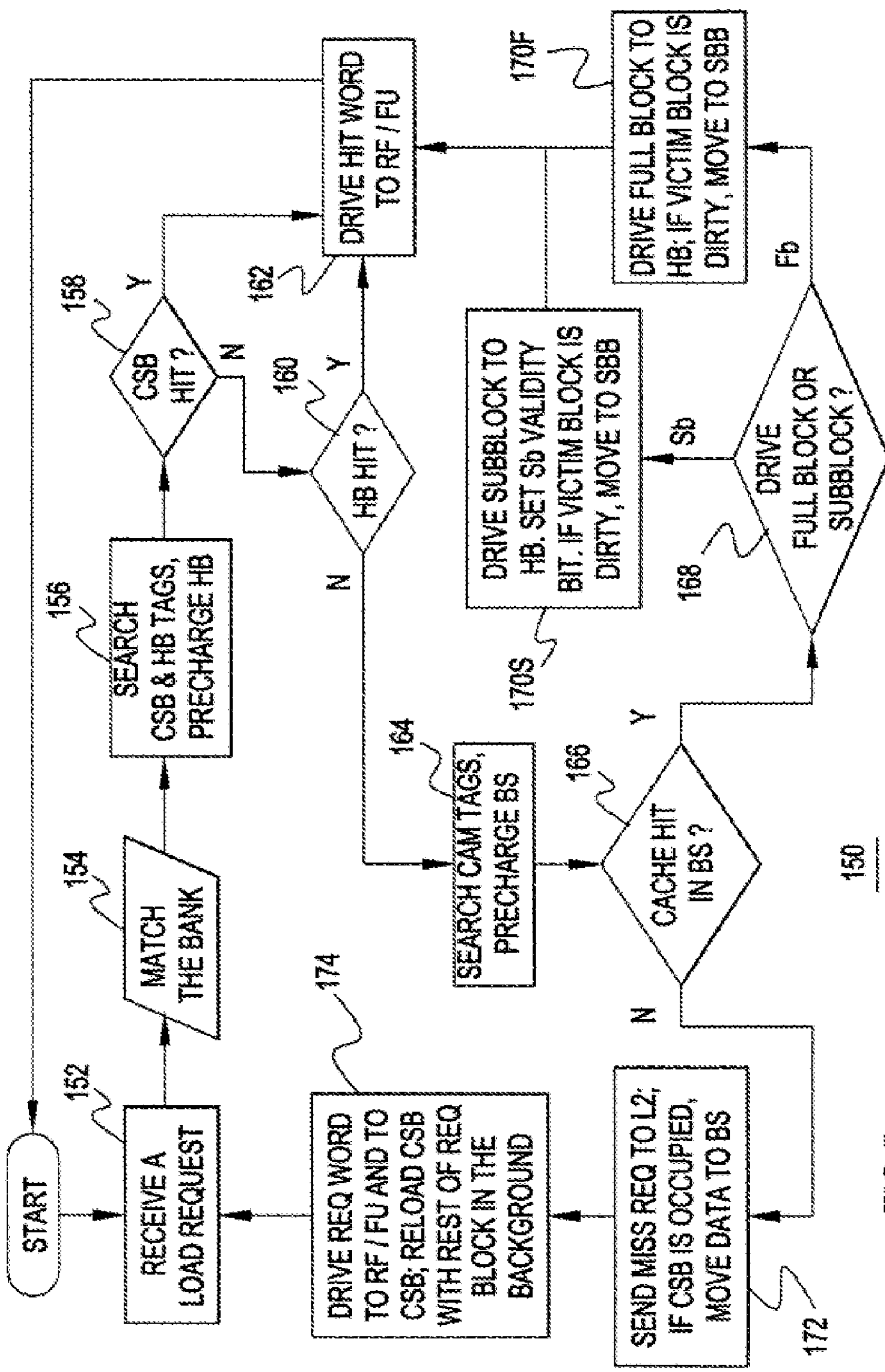

FIG.5

EXHIBIT F

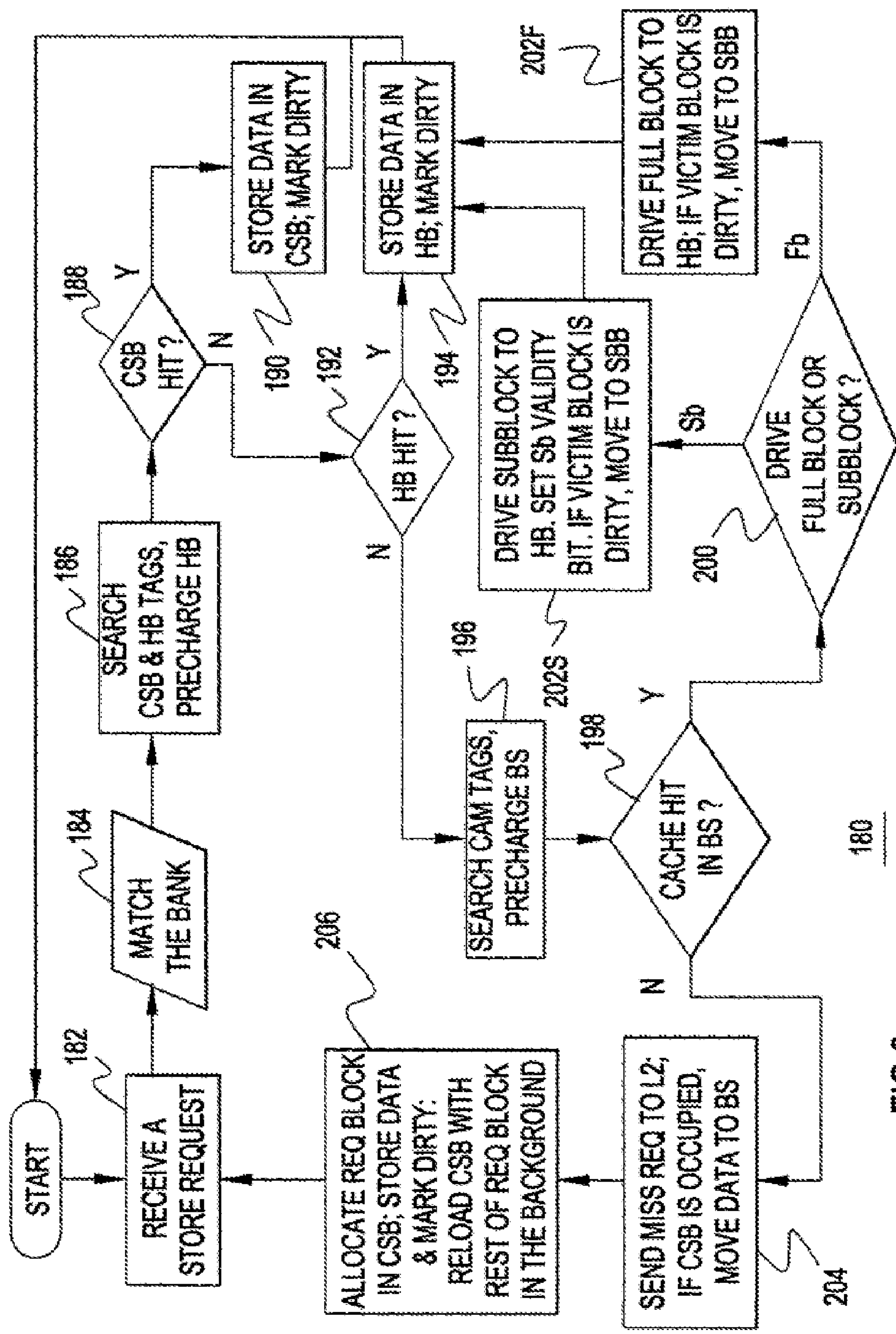

FIG.6

EXHIBIT G

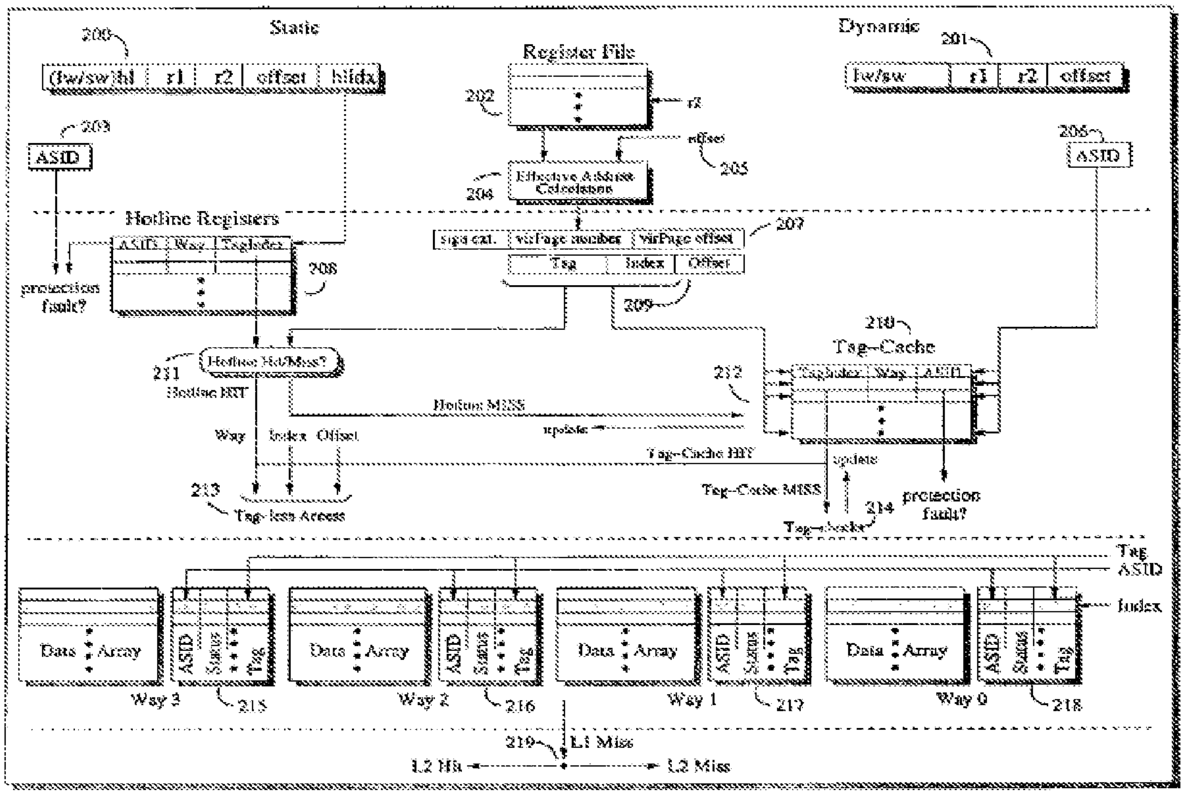

Fig. 4

RELATED PROCEEDINGS APPENDIX

Pursuant to 35 U.S.C. §41.37(c)(x), copies of the following decisions rendered by a court of the Board in any proceeding identified above under 35 U.S.C. §41.37(c)(1)(ii) are enclosed herewith. As appellants are aware no decisions or proceedings having a bearing on the present appeal, nothing is included in the Appendix.